Evolutionary Multiobjective Design targeting a Field Programmable Transistor Array

Arturo Hernández Aguirre
Center for Research in Mathematics
Guanajuato, 36240, México
artha@cimat.mx

Ricardo S. Zebulum
Jet Propulsion Laboratory
Pasadena, CA 91109 USA
ricardo.s.zebulum@jpl.nasa.gov

Carlos Coello Coello
CINVESTAV-IPN, EE Section
México, D.F. 07300 México
ccoello@cs.cinvestav.mx

Abstract- This paper introduces the ISPAES algorithm for circuit design targeting a Field Programmable Transistor Array (FPTA). The use of evolutionary algorithms is common in circuit design problems, where a single fitness function drives the evolution process. Frequently, the design problem is subject to several goals or operating constraints, thus, designing a suitable fitness function catching all requirements becomes an issue. Such a problem is amenable for multi-objective optimization, however, evolutionary algorithms lack an inherent mechanism for constraint handling. This paper introduces ISPAES, an evolutionary optimization algorithm enhanced with a constraint handling technique. Several design problems targeting a FPTA show the potential of our approach.

1 Introduction

The success of Evolutionary Algorithms (EAs) in global optimization has triggered a considerable amount of research whose goal is a mechanisms to handle constraints [8]. So far, the most common approach adopted in the evolutionary optimization literature to deal with constrained search spaces is the use of penalty functions [11]. Despite the popularity of penalty functions, they have several drawbacks from which the main one is that they require a careful fine tuning of the penalty factors that indicates the degree of penalization to be applied [12].

Recently, some researchers have suggested the use of multiobjective optimization concepts to handle constraints in EAs. This paper introduces a new approach that is based on an evolution strategy that was originally proposed for multiobjective optimization: the Pareto Archived Evolution Strategy (PAES) [7]. ISPAES (Inverted Shrinkable PAES) can be used to handle constraints in single-objective optimization problems and does not present the scalability problems of the original PAES. Besides using Pareto-based selection, our approach uses a secondary population (one of the most common notions of elitism in evolutionary multiobjective optimization), and a mechanism that reduces the constrained search space so that our technique can approach a optimum more efficiently.

Evolvable hardware researchers apply EAs in circuit design, usually in a highly dimensional search space, where building a solution is more promissory than deriving it from the premises. The common approach can be defined as global optimization, that is, the objective function is represented by the fitness function, and no problem constraints are present. In some problems, a simple form of constraint handling (for instance penalty functions), is applied to the population in order to deal with constrained search spaces.

In this paper, the ISPAES algorithm is applied to circuit design over a Field Programmable Transistor Array model. The FPTA transistor model and circuit description was fed into the SPICE simulator for all experiments, so "extrinsic evolution" was used. Experiments using the real FPTA device have been reported by Adrian Stoica et al. [13, 14]. His technique, however, is driven by the optimization of the sole fitness function.

2 Problem Statement

We are interested in the general non-linear programming problem in which we want to:

$$\text{Find } \mathbf{x} \text{ which optimizes } \mathbf{F}(\mathbf{x}) \tag{1}$$

subject to:

$$g_i(\mathbf{x}) \le 0, \;\; i = 1, \ldots, n \tag{2}$$

$$h_j(\mathbf{x}) = 0, \;\; j = 1, \ldots, p \tag{3}$$

where $\mathbf{F}$ is the vector of objective function values $\mathbf{F} = [f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})]$, $\mathbf{x}$ is the vector of solutions $\mathbf{x} = [x_1, x_2, \ldots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or non-linear).

If we denote the feasible region with $\mathcal{F}$ and the entire search space with $\mathcal{S}$, then obviously $\mathcal{F} \subseteq \mathcal{S}$.

For an inequality constraint that satisfies $g_i(\mathbf{x}) = 0$, then we will say that it is active at $\mathbf{x}$. All equality constraints h_j (regardless of the value of $\mathbf{x}$) are considered to be active at all points of $\mathcal{F}$.

An optimality criteria has to be defined for the multiobjective problems since decisions of the type *what is the best solution?* must be taken over the population individuals. Pareto dominance is that criteria; we say one individual dominates a second individual if the first is better in at least one of the objectives while the other objectives remain with no change (a$\preceq$b means a dominates b).

3 Basic Concepts

A multiobjective optimization problem (MOP) has the following the form:

$$\text{Minimize } [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \tag{4}$$

Proceedings of the 2004 NASA/DoD Conference on Evolution Hardware (EH'04)
0-7695-2145-2/04 $ 20.00 © 2004 IEEE

subject to the m inequality constraints:

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, \ldots, m \tag{5}$$

and the p equality constraints:

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{6}$$

where k is the number of objective functions $f_i : R^n \to R$. We call $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ the vector of decision variables. We wish to determine from among the set $\mathcal{F}$ of all vectors which satisfy (5) and (6) the particular set of values $x_1^*, x_2^*, \ldots, x_n^*$ which yield the optimum values of all the objective functions.

3.1 Pareto Optimality

A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if u is partially less than v, i.e., $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$. For a given multiobjective optimization problem, $\vec{f}(x)$, the Pareto optimal set ($\mathcal{P}^*$) is defined as:

$$\mathcal{P}^* := \{x \in \mathcal{F} \mid \neg\exists\, x' \in \mathcal{F} \;\; \vec{f}(x') \preceq \vec{f}(x)\}. \tag{7}$$

Thus, we say that a vector of decision variables $\vec{x}^* \in \mathcal{F}$ is *Pareto optimal* if there does not exist another $\vec{x} \in \mathcal{F}$ such that $f_i(\vec{x}) \leq f_i(\vec{x}^*)$ for all $i = 1, \ldots, k$ and $f_j(\vec{x}) < f_j(\vec{x}^*)$ for at least one j. In words, this definition says that $\vec{x}^*$ is Pareto optimal if there exists no feasible vector of decision variables $\vec{x} \in \mathcal{F}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. Unfortunately, this concept almost always gives not a single solution, but rather a set of solutions called the *Pareto optimal set*. The vectors $\vec{x}^*$ correspoding to the solutions included in the Pareto optimal set are called *nondominated*. The image of the Pareto optimal set under the objective functions is called *Pareto front*.

4 Related Work

The main idea of adopting multiobjective optimization concepts to handle constraints is to redefine the single-objective optimization of $f(\vec{x})$ as a multiobjective optimization problem in which we will have $m + 1$ objectives, where m is the total number of constraints. Then, we can apply any multiobjective optimization technique [5] to the new vector $\bar{v} = (f(\vec{x}), f_1(\vec{x}), \ldots, f_m(\vec{x}))$, where $f_1(\vec{x}), \ldots, f_m(\vec{x})$ are the original constraints of the problem. An ideal solution $\vec{x}$ would thus have $f_i(\vec{x})$=0 for $1 \leq i \leq m$ and $f(\vec{x}) \leq f(\vec{y})$ for all feasible $\vec{y}$ (assuming minimization).

These are the mechanisms taken from evolutionary multiobjective optimization that are more frequently incorporated into constraint-handling techniques:

1. Use of Pareto dominance as a selection criterion.

2. Use of Pareto ranking [6] to assign fitness in such a way that nondominated individuals (i.e., feasible individuals in this case) are assigned a higher fitness value.

3. Split the population in subpopulations that are evaluated either with respect to the objective function or with respect to a single constraint of the problem.

In order to sample the feasible region of the search space widely enough to reach the global optima it is necessary to maintain a balance between feasible and infeasible solutions. If this diversity is not reached, the search will focus only on one area of the feasible region. Thus, it will lead to a local optima solution.

A multiobjective optimization technique aims to find a set of trade-off solutions which are considered good in all the objectives to be optimized. In global nonlinear optimization, the main goal is to find the global optimum. Therefore, some changes must be done to those approaches in order to adapt them to the new goal. Our main concern is that feasibility takes precedence, in this case, over nondominance. Therefore, good "trade-off" solutions that are not feasible cannot be considered as good as bad "trade-off" solutions that are feasible. Furthermore, a mechanism to maintain diversity must normally be added to any evolutionary multiobjective optimization technique. In our proposal, diversity is kept by using an adaptable grid, and by a selection process applied to the external file that maintains a mixture of both good "trade-off" and feasible individuals.

There are several approaches that have been developed using multiobjective optimization concepts to handle constraints, but due to space limitations we will not discuss them here (see for example [4, 15, 9, 10]).

Evolutionary multiobjective optimization has been applied to the synthesis of low-power operational amplifiers [16, 17]. The approach, however, is based on Genetic Algorithms.

5 ISPAES Algorithm

ISPAES (Inverted Shrinkable PAES) [2], uses Pareto dominance as the criterion selection, but unlike the previous work in the area, a secondary population is used in this case. The approach, which is a relatively simple extension of PAES [7] provides, however, very good results, which are highly competitive with those generated with an approach that represents the state-of-the-art in constrained evolutionary optimization. The structure of ISPAES algorithm is shown in Figure 1. Notice the two loops operating over the Pareto set (in the external storage). The right loop aims for exploration of the search space, the left loop aims for population diversity and exploitation.

ISPAES has been implemented as an extension of the Pareto Archived Evolution Strategy (PAES) proposed by Knowles and Corne [7] for multiobjective optimization. PAES's main feature is the use of an adaptive grid on which objective function space is located using a coordinate system. Such a grid is the diversity maintenance mechanism of PAES and it constitutes the main feature of this algorithm. The grid is created by bisecting k times the function space of dimension d (d is the number of objective functions of the problem. In our case, d is given by the total number of constraints plus one. In other words, $d = n + p + 1$,

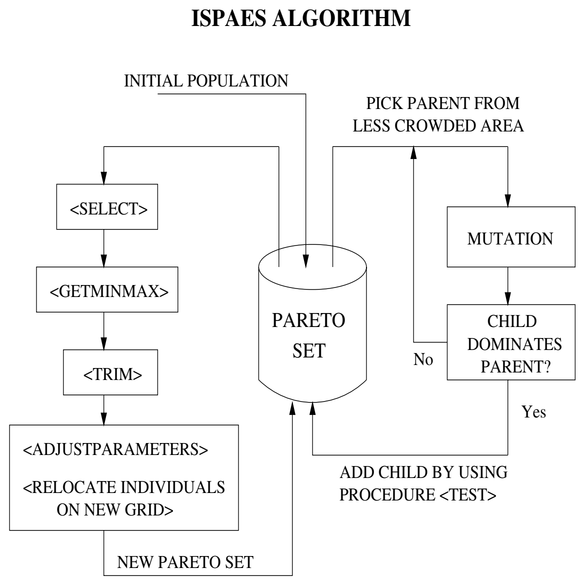

Figure 1: The logical structure of ISPAES algorithm

$maxsize$: max size of Pareto store
$maxffeval$: fitness function evaluations
Initialize Pareto store with maxsize individuals
While gen$\leq$MaxGen **do**
Pick μ parents from less crowded area
Run $(\mu + \lambda)$-ES until maxffeval is met
test(Pareto store, λ children)
test: adds children to Pareto store
shrinkspace(Pareto store): reduce search space
End While

Figure 2: Main algorithm of our $ISPAES$

if (current $<$ maxsize) **then** {
add(h);
if (h $\square$ c) **then** c=h }
else if ($\exists a_p \in$file | h $\square$ a_p) **then** {
remove(a_p); add(h)
if (h $\square$ c) **then** c = h; }

Figure 3: Pseudo-code of **test(h,c,file)** (called by **main** of ISPAES)

where n is the number of inequality constraints, and p is the number of equality constraints. Note that we add one to this summation to include the original objective function of the problem). The control of 2^{kd} grid cells means the allocation of a large amount of physical memory for even small problems. For instance, 10 functions and 5 bisections of the space produce 2^{50} cells. Thus, the first feature introduced in ISPAES is the "inverted" part of the algorithm that deals with this space usage problem. ISPAES's fitness function is mainly driven by a feasibility criterion. Global information carried by the individuals surrounding the feasible region is used to concentrate the search effort on smaller areas as the evolutionary process takes place. In consequence, the search space being explored is "shrunk" over time. Eventually, upon termination, the size of the search space being inspected will be very small and will contain the solution desired (in the case of single-objective problems. For multi-objective problems, it will contain the feasible region).

The main algorithm of ISPAES is shown in Figure 2. Its goal is the construction of the Pareto front which is stored in an external memory (called *file*). The algorithm performs *Maxnew* loops, generating a child *h* from a random parent *c* in every loop. Therefore, the ISPAES algorithm introduced here is based on a $ES(1 + 1)$. If the child is better than the parent, that is, the child dominates its parent, then it is inserted in *file*, and its position is recorded. A child is generated by introducing random mutations to the parent, thus, $h = mutate(c)$ will alter a parent with increments whose standard deviation is governed by Equation 9.

Most of **main** and the function **test(h,c,file)** in ISPAES are devoted to three things: (1) decide whether a new child should be inserted in *file*, and if so, (2) how to make room for the new member and (3) who becomes the new parent. Every *g* new children created, a new parent is randomly picked from *file* for this purpose. Also, every *r* children generated, the space is shrunk around the current Pareto front represented by the individuals of the external memory. Here we introduce the following notation: $x_1 \square x_2$ means x_1 is located in a less populated region of the grid than x_2. The pseudo-code of this function is depicted in Figure 3.

5.1 Inverted "ownership"

PAES keeps a log of every greed location, whereas ISPAES keeps a log of the position of every individual. The advantage of this inverted relationship is clear when the optimization problem has many functions (more than 10), and/or the granularity of the grid is fine, for in this case only ISPAES is able to deal with any number of functions and granularity level.

5.2 Shrinking the Objective Space

Shrinkspace(file) is the most important function of ISPAES since its task is the reduction of the search space. The space is reduced every r number of generations. The pseudo-code of **Shrinkspace(file)** is shown in Figure 4.

In the following we describe the four tasks performed by **shrinkspace**.

- The function **select(file)** returns a list whose elements are the best individuals found in *file*. The size of the list is set to 15% of *maxsize*. Thus, the goal of **select(file)** is to create a list with: a) only the best feasible individuals, b) a combination of feasible and partially feasible individuals, or c) the "most promis-

$\underline{\boldsymbol{x}}_{pob}$: vector containing the smallest value of either $x_i \in X$
$\overline{\boldsymbol{x}}_{pob}$: vector containing the largest value of either $x_i \in X$
select(file);
getMinMax(file, $\underline{\boldsymbol{x}}_{pob}$, $\overline{\boldsymbol{x}}_{pob}$);
trim($\underline{\boldsymbol{x}}_{pob}$, $\overline{\boldsymbol{x}}_{pob}$);
adjustparameters(file);

Figure 4: Pseudo-code of **Shrinkspace(file)** (called by **main** of IS-PAES)

```
m: number of constraints
i: constraint index
maxsize: max size of file
listsize: 50% of maxsize
constraintvalue(x,i): value of individual at constraint i
sortfile(file): sort file by objective function
worst(file,i): worst individual in file for constraint i
validconstraints={1,2,3,...,m};
i=firstin(validconstraints);
While (size(file) > listsize and size(validconstraints) > 0) {
    x=worst(file,i)
    if (x violates constraint i)
        file=delete(file,x)
    else validconstraints=removeindex(validconstraints,i)
if (size(validconstraints) > 0) i=nextin(validconstraints)
}
if (size(file) == listsize))
    list=file
else
    file=sort(file)
    list=copy(file,listsize) *pick the best listsize elements*
```

Figure 5: Pseudo-code of **select(file)** (called by **shrinkspace**)

ing” infeasible individuals. The selection algorithm is shown in Figure 5. Note that $validconstraints$ (a list of indexes to the problem constraints) indicates the order in which constraints are tested. The loop steps over the constraints removing only one (the worst) individual for each constraint till there is none to delete (all feasible), or 15% of file size is reached (in other words, 85% of the Pareto set will be generated anew using the best 15% individuals as parents). Also, in order to keep diversity, a new parent is randomly chosen from the less populated region of the grid after placing on it g new individuals.

- The function **getMinMax(file)** takes the list *list* (last step in Figure 5) and finds the extreme values of the decision variables represented by those individuals. Thus, the vectors $\underline{\boldsymbol{x}}_{pob}$ and $\overline{\boldsymbol{x}}_{pob}$ are found.
- Function **trim**($\underline{\boldsymbol{x}}_{pob}$, $\overline{\boldsymbol{x}}_{pob}$) shrinks the feasible space around the potential solutions enclosed in the hypervolume defined by the vectors $\underline{\boldsymbol{x}}_{pob}$ and $\overline{\boldsymbol{x}}_{pob}$. Thus, the function **trim**($\underline{\boldsymbol{x}}_{pob}$, $\overline{\boldsymbol{x}}_{pob}$) (see Figure 6) determines the new boundaries for the decision variables.

The value of β is the percentage by which the boundary values of either $x_i \in X$ must be reduced such that the resulting hypervolume H is a fraction α of its previous value. The function **trim** first finds in the population the boundary values of each decision variable: $\overline{x}_{pob,i}$ and $\underline{x}_{pob,i}$. Then the new vectors $\overline{x}_i$ and $\underline{x}_i$ are updated by $deltaMin_i$, which is the reduction in each variable that in the overall reflects a change in the volume by a factor β. In ISPAES all objective variables are reduced at the same rate β, therefore, β can be deduced from α as discussed next. Since we need the new hypervolume be a fraction α of the previous one,

$$H_{\text{new}} \geq \alpha H_{\text{old}} \tag{8}$$

n: size of decision vector;
$\overline{x}_i$: actual upper bound of the i_{th} decision variable
$\underline{x}_i$: actual lower bound of the i_{th} decision variable
$\overline{x}_{pob,i}$: upper bound of i_{th} decision variable in population
$\underline{x}_{pob,i}$: lower bound of i_{th} decision variable in population
$\forall i$: i $\in$ { 1, ..., n }
    $slack_i = 0.05 \times (\overline{x}_{pob,i} - \underline{x}_{pob,i})$
    $width\_pob_i = \overline{x}_{pob,i} - \underline{x}_{pob,i}$; $width_i^t = \overline{x}_i^t - \underline{x}_i^t$
    $deltaMin_i = \frac{\beta * width_i^t - width\_pob_i}{2}$
    delta$_i$ = max(slack$_i$, deltaMin$_i$);
    $\overline{x}_i^{t+1} = \overline{x}_{pob,i} + delta_i$; $\underline{x}_i^{t+1} = \underline{x}_{pob,i} - delta_i$;
    if ($\overline{x}_i^{t+1} > \overline{x}_{original,i}$) **then**
        $\underline{x}_i^{t+1} -= \overline{x}_i^{t+1} - \overline{x}_{original,i}$; $\overline{x}_i^{t+1} = \overline{x}_{original,i}$;
    if ($\underline{x}_i^{t+1} < x_{original,i}$) **then** $\overline{x}_i^{t+1} += \underline{x}_{original,i} - \underline{x}_i^{t+1}$;
        $\underline{x}_i^{t+1} = \underline{x}_{original,i}$;
    if ($\overline{x}^{t+1} > \overline{x}_{original,i}$) **then** $\overline{x}_i^{t+1} = \overline{x}_{original,i}$;

Figure 6: Pseudo-code of **trim** (called by **shrinkspace**)

$$\prod_{i=1}^{n}(\overline{x}_i^{t+1} - \underline{x}_i^{t+1}) = \alpha \prod_{i=1}^{n}(\overline{x}_i^t - \underline{x}_i^t)$$

Either x_i is reduced at the same rate β, thus

$$\prod_{i=1}^{n}\beta(\overline{x}_i^t - \underline{x}_i^t) = \alpha \prod_{i=1}^{n}(\overline{x}_i^t - \underline{x}_i^t)$$
$$\beta^n \prod_{i=1}^{n}(\overline{x}_i^t - \underline{x}_i^t) = \alpha \prod_{i}^{n}(\overline{x}_i^t - \underline{x}_{i=1}^t)$$
$$\beta^n = \alpha$$
$$\beta = \alpha^{\frac{1}{n}}$$

In short, the new search interval of each decision variable x_i is adjusted as follows (the complete algorithm is shown in Figure 4):

$$width_{new} \geq \beta \times width_{old}$$

It should be noted that the value of α has an important impact on the performance of ISPAES because it controls the shrinking speed. In order to determine a range within which we could set this parameter for a large variety of problems, we studied the effect of α on the performance of our algorithm for many test problems. From analyzing this effect, we found that in all cases, a range of α between 85% and 97% was always able to generate the best possible solutions to each problem. Values smaller than 0.80 make the algorithm prone to converge to local minima. Values of α too near to 100% slow down convergence, although they increase the probability of success. In order to avoid a fine tuning of α dependent of each test function, we decided to set its value to 0.90, which we considered as a good compromise based on our analysis. As we will see later on, this value of α provided good results in all the problems solved.

Note that also the parameter r (see Figure 2), which controls the shrinkspace rate, plays an important role in the algorithm. To set the value of r, we performed a similar analysis to the one previously described for α.

In this analysis, we related the behavior of r with that of α and with the performance of ISPAES. Our results indicated that a value of $r = 2 * maxsize$ provided convergence to the optimum in most of the problems ($maxsize$ is the number of elements allowed to the Pareto set, stored in the external file). Thus, we used $r = 200$, and $maxsize = 100$ in all the experiments reported in this paper.

The variable *slack* is calculated once every new search interval is determined (usually set to 5% of the interval). The role of slack is simply to prevent (up to some extend) against fast decreasing rates of the search interval.

- The last step of **shrinkspace()** is a call to **adjustparameters(file)**. The goal is to re-start the control variable σ through:

$$\sigma_i = (\overline{x}_i - \underline{x}_i)/\sqrt{n} \quad i \in (1, \ldots, n) \tag{9}$$

This expression is also used during the generation of the initial population. In that case, the upper and lower bounds take the initial values of the search space indicated by the problem. The variation of the mutation probability follows the exponential behavior suggested by Bäck [3].

Elitism
A special form of elitism is implemented in IS-PAES to prevent the lost of the best individual. Elitism is implemented as follows: the best individual of the generation is marked and only replaced by another one if it is in the feasible region and with better objective function value.

ISPAES for Optimizing problems in Discrete Search Space
Simple modifications are required for discrete optimization problems. The initial value of all objective variables is a random integer drawn from an uniform distribution, and bounded by the upper and lower limits staten by the specific problem.
Mutation of objective variables is performed as follows,

$$x_i^{t+1} = x_i^t + rand(\sigma_i)$$

where σ_i is the control variable of the corresponding objective variable, and $rand(\sigma_i)$ is a random number with uniform distribution in the interval $[0, \sigma]$.
Control variables σ_i are mutated as follows,

$$if(random() < 0.45) \ \ then\, \sigma = \sigma + 1; \ \ else\, \sigma = \sigma - 1;$$

this is, with little less probability than the average of 0.5, the control variables diminish their value by 1.
The reduction of the search space is performed as shown in Figure 6 for the real space case, except that all results of the computations must be rounded up to the next integer. The variable $slack$ is also computed as depicted in Figure 6, it must also be rounded up, and its smallest possible value is 1.

Input Ramp	GA	ISPAES
1.0v	2.2963v	2.3711v
2.5v	2.2246v	2.2614v

Table 1: Output voltages of NOT-gate synthesized by ISPAES and a GA.

6 Experiments

6.1 The targeted FPTA

The FPTA model used in this paper can be described as a configurable operational amplifier. Every configuration is made by setting 77 switches, therefore, there are 2^{77} possible circuits in the search space. Several input and output pads are available to feed in and read data back. For the following experiments we could simply assume there is one input and a couple of outputs.

6.2 NOT-Gate design

The goal of this experiment is to get two "digital" levels for an input ramp. The ramp starts at 1.0v, and finishes at 2.5v We did use a Pareto set of 150 individuals, and 200 generations. Every 50 generations the members of Pareto set are recalculated after a "trim" operation. (note: for ISPAES all constraints and objective function are treated as a minimization problem. So, the goal and constraints are restated as a minimization case)

One objective is declared as follows:

$$O_1 = OutVoltage_{input=2.5} - OutVoltage_{input=1.0} < 0 \tag{10}$$

The rationale behind Equation 10 is that the more negative the greater the swing between output voltages. There are no more requirements over the outputs, other than the maximum swing. Only one constraint is defined to ISPAES; the goal is to penalize circuits that SPICE finds "impossible" to evaluate, or for which it generates out of scale voltages (the reason is also the impossibility of evaluation). Thus, correct circuits get a constraint value of -1 (feasible solution), and invalid circuits a constraint value of 10 (infeasible solution). The output voltages of the NOT gate are shown in Table 1. Notice that the output voltage swing is about 53% greater in the circuit found by ISPAES (constrained optimization) than the GA approach (unconstrained optimization) [1].

6.3 NOT-gate for 4-steps ramp

In preparation for the 2-bit ADC design example, a NOT-gate is designed here to respond to a input ramp of 4 steps. The ramp starts at 1.5v with step size of 0.5v The required responses are the logic voltages for "0-1-0-1". We did use a Pareto set of 150 individuals, and 200 generations. Every 50 generations the members of Pareto set are recalculated after a "trim" operation.

This design problem is restated by using 1 objective function plus 5 constraints. One of those constraints is used to penalize impossible circuits, as described in the previous example. The objective function is also driven by the

Input Ramp	GA	ISPAES
1.5v	2.6571v	1.9601v
2.0v	2.7081v	3.0914v
2.5v	2.6700v	1.9394v
3.0v	2.7198v	3.1650v

Table 2: Output voltages of NOT-gate for 4-steps ramp synthesized by ISPAES and a GA.

maximization of the difference between low and high output levels, as follows.

$$O_1 = OV_{in=1.5v} + OV_{in=2.5v} - OV_{in=2.0v} - OV_{in=3.0v} < 0 \quad (11)$$

In Equation 11, OV means “output voltage”. Four constraints are added to the requirements, the goal is to drive the output low levels below 2.0v, and the output high levels above 2.0v, as follows.

$$\begin{aligned} C_1 &= OV_{in=1.5v} - 2.0 < 0 && (12)\\ C_2 &= OV_{in=2.5v} - 2.0 < 0 && (13)\\ C_3 &= 2 - OV_{in=2.0v} < 0 && (14)\\ C_4 &= 2 - OV_{in=3.0v} < 0 && (15) \end{aligned}$$

The output voltages for this problem are shown in Table 2. Note the difference between voltages found by the unconstrained (GA) and the constrained evolutionary optimization approach (ISPAES).

6.4 2-bit ADC

The last experiment is the design of a 2-bit ADC. The four steps input ramp start at 1.5v, step size of 0.5v, so final ramp value is 3.0v The two output values are named MSB and LSB (most and least significant bit). We did use a Pareto set of 150 individuals, and run for 1000 generations. Every 50 generations the members of Pareto set are recalculated after a “trim” operation.

This problem is restated as a 12 objective problem in the following way: 2 objectives, 4 constraints on the MSB, 4 constraint on the LSB, 2 constraints for incorrect circuits (one on each output). The objectives are introduced as shown in Equations 16 and 17.

$$\begin{aligned} O_1^{LSB} &= OV_{in=1.5v}^{LSB} + OV_{in=2.5v}^{LSB} \\ &\quad - OV_{in=2.0v}^{LSB} - OV_{in=3.0v}^{LSB} < 0 && (16)\\ O_2^{MSB} &= OV_{in=1.5v}^{MSB} + OV_{in=2.5v}^{MSB} \\ &\quad - OV_{in=2.0v}^{MSB} - OV_{in=3.0v}^{MSB} < 0 && (17) \end{aligned}$$

The constraints over voltage levels are similar to those used in the previous experiment, so for each output we look for low levels below 2.0v and high levels above 2.0v.

$$C_1 = OV_{in=1.5v}^{LSB} - 2.0 < 0 \quad (18)$$

Input Ramp	GA-LSB	ISPAES-LSB
1.5v	1.9311v	1.9370v
2.0v	1.9512v	2.015v
2.5v	1.9446v	1.3323v
3.0v	2.0813v	3.1183v

Table 3: Output voltages of LSB for 2-bit ADC synthesized by ISPAES and a GA.

Input Ramp	GA-MSB	ISPAES-MSB
1.5v	2.1522v	1.6558v
2.0v	2.4962v	1.7497v
2.5v	2.8645v	2.3604v
3.0v	3.2514v	2.6551v

Table 4: Output voltages of MSB for 2-bit ADC synthesized by ISPAES and a GA.

$$\begin{aligned} C_2 &= OV_{in=2.5v}^{LSB} - 2.0 < 0 && (19)\\ C_3 &= 2 - OV_{in=2.0v}^{LSB} < 0 && (20)\\ C_4 &= 2 - OV_{in=3.0v}^{LSB} < 0 && (21)\\ C_5 &= OV_{in=1.5v}^{MSB} - 2.0 < 0 && (22)\\ C_6 &= OV_{in=2.0v}^{MSB} - 2.0 < 0 && (23)\\ C_7 &= 2 - OV_{in=2.5v}^{MSB} < 0 && (24)\\ C_8 &= 2 - OV_{in=3.0v}^{MSB} < 0 && (25) \end{aligned}$$

As stated before, 2 more constraints were used to penalize “impossible” circuits (in the same way as described in previous experiments).

Output voltages for LSB and MSB are reported in Tables 3 and 4. Note that for ISPAES is natural to deal with design requirements (constraints), for instance, the common voltage level of 2.0v set for the outputs.

7 Final Remarks and Conclusions

Many circuit design problems are amenable for multiobjective optimization since all requirements can be incorporated into the design process. EAs are proper tools for unconstrained optimization but, specialized EAs, thus, augmented with a constraint handling technique, are needed to deal with constrained problems. The easy way to deal with constraints, for instance, penalization, becomes weaker as a problem grows. Thus, it is rather interesting and necessary to redefine the optimality concept in the presence of several goal and constraints. The ISPAES method just introduced, proposes a technique to handle constraints based on Pareto dominance. Even more, the main contribution is an unbiased selection operator that picks feasible and unfeasible individuals, thus keeping diversity during the evolutionary process.

The experiments performed are simple but results already show the advantages of multiobjective optimization techniques for evolutionary circuit synthesis. The benefits are unquestionable, for instance, the introduction of required output voltage levels, frequency response, time response, power consumption, etc.

Acknowledgments

The first author acknowledge support from CONACyT through projects P40721-Y and 42523. The third author acknowledges support from CONACyT through project number 42435-Y. Thanks to Ibrahim Gokcen for coding some portions of the system's interface.

Bibliography

[1] Arturo Hernández Aguirre. Evolutionary Circuit Design on the FPTA-2. Technical Report Contract No. 1230282, Jet Propulsion Laboratory, Pasadena, CA, July 2001.

[2] Arturo Hernández Aguirre, S. Botello, C. Coello, and G. Lizárraga. Use of Multiobjective Optimization Concepts to Handle constraints in Single Objective Optimization. In Erick Cantú Paz, editor, *Genetic and Evolutionary Computation - GECCO 2003*, volume 1, pages 573–584, Chicago,IL, USA, July 2003. Springer-Verlag. Lecture Notes in Computer Science No. 2723.

[3] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[4] Eduardo Camponogara and Sarosh N. Talukdar. A Genetic Algorithm for Constrained and Multiobjective Optimization. In Jarmo T. Alander, editor, *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, pages 49–62, Vaasa, Finland, August 1997. University of Vaasa.

[5] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[6] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[7] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[8] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[9] I. C. Parmee and G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pages 97–102, Plymouth, UK, 1994. University of Plymouth, University of Plymouth.

[10] Tapabrata Ray, Tai Kang, and Seow Kian Chye. An Evolutionary Algorithm for Constrained Optimization. In Darrell Whitley et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, 2000. Morgan Kaufmann.

[11] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some Guidelines for Genetic Algorithms with Penalty Functions. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, pages 191–197, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.

[12] Alice E. Smith and David W. Coit. Constraint Handling Techniques—Penalty Functions. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.

[13] Adrian Stoica, R. Zebulum, D. Keymeulen, R. Tawel, T. Daud, and A. Thakoor. Reconfigurable VLSI Architectures for Evolvable Hardware: from Experimental Field Programmable Transistor Arrays to Evolution-Oriented Chips. *IEEE Transactions of VLSI Systems, Special Issue on Reconfigurable and Adaptive VLSI Systems*, 9(1):227–232, 2001.

[14] Adrian Stoica, R. Zebulum, M.I.Ferguson, D. Keymeulen, V. Duong, T. Daud, and Xin Guo. Evolutionary Configuration of Field Programmable Analog Devices. In *Proceedings of the IEEE AeroSpace Conference*. IEEE, IEEE Press, March 2003.

[15] Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3):391–412, 1997.

[16] R. S. Zebulum, M. A. Pacheco, and M. Vellasco. A multi-objective optimisation methodology applied to the synthesis of low-power operational amplifiers. In Ivan Jorge Cheuri and Carlos Alberto dos Reis Filho, editors, *Proceedings of the XIII International Conference in Microelectronics and Packaging*, volume 1, pages 264–271, Curitiba, Brazil, August 1998.

[17] R. S. Zebulum, M. A. Pacheco, and M. Vellasco. Synthesis of CMOS operational amplifiers through Genetic Algorithms. In *Proceedings of the Brazilian Symposium on Integrated Circuits, SBCCI'98*, pages 125–128, Rio de Janeiro, Brazil, September 1998. IEEE.